

Monograf as Nuevas Publicar Blogs Foros

Busqueda avanzada

B 

Monograf as.com > Computacion > Programacion

  Descargar

Imprimir

Comentar

Ver trabajos relaci

El modelo de mccall como aplicaci n de la calidad a la revision del software de gestion empresarial

Enviado por [angel.cervera](#)

8+1 0

Twitter 0

Me gusta 3

Darby

  darbysoft.com

Software de gesti n, presupuestos costes y proyectos

1. Resumen

2. Abstract

4. Calidad: definiciones.

5. Calidad del software.

6. La calidad a trav s de la normalizaci n en la ingenier a del software y su problem tica.

7. Los modelos de calidad del software de gesti n.

8. El modelo de McCall.

9. C mo emplear el modelo de mccall.

10. Conclusiones.

11. Bibliografia.

1. Resumen

La mayor importancia de las nuevas tecnolog as de la informaci n y su creciente presencia en los diversos  mbitos de la industria moderna (robots, centros de control, etc.) y sus productos finales (aviaci n, autom viles, electrodom sticos, telefon a, etc.) conlleva cada vez m s la presencia de programas inform ticos q gobiernan muchas de sus prestaciones, o bien como herramientas que el cliente emplear  en su propio beneficio. Basta con observar la fren tica actividad que h supuesto para la industria, las empresas de servicios y la Administraci n el enfrentarse al tan temido "efecto 2000", as  como el gasto que ha conllevado la revi modificaci n de los programas, para vislumbrar la punta de un iceberg: la falta de un control riguroso y sistem tico de la calidad del software de gesti n. En el presente trabajo se aborda este tema y se presenta un modelo de aplicaci n que ayudaría a proveedores y clientes desde el comienzo del dise o de una aplicaci n espec fica de software para su negocio o actividad.

Palabras Claves: calidad, hardware, modelo de McCall, software.

2. Abstract

The major importance of new information technologies and their increasing presence in the various scopes of modern industry (robots, control centres, etc.) to with their end products (aviation, automobiles, household appliances, telephony, etc.) is increasingly leading to a greater presence of computer programs either controlling many of its features, or as tools that the client will use in his own benefit. You only need to observe the frenetic activity that facing the fearful Y2F has supposed for industry, service companies and Administration as well as the expenses brought forth by the revision and modification of programs, in order t catch a glimpse of the tip of the iceberg: the lack of a rigorous and systematic control of the quality of management software. In the present work this subject i approached and an application model is depicted that would help suppliers and clients from the very first stages of the design of a specific software application their business or activity.

Keywords: Quality, hardware, McCall's model, software.

3. Algunos antecedentes al concepto de calidad.

A lo largo de toda la historia la b squeda y el af n de perfecci n por parte del hombre ha sido constante, de tal forma, que el inter s por el trabajo bien hecho y necesidad de asumir responsabilidades sobre la labor efectuada poco a poco deriv  en el concepto de calidad.

Un ejemplo temprano se encuentra entre los 2000 y 3000 a os A.C. cuando los faraones egipcios mandaron construir las famosas pir mides de Egipto. Muchas ellas tienen par metros que las acercan casi a la perfecci n en la construcci n pues en la orientaci n de la base con respecto a la alineaci n N-S, E-W el error n llega a ser de 6 minutos de arco, distando la base de algunas de ellas de ser un cuadrado perfecto menos de 17,78 cm. Todo ello se consegu a gracias a los m tc inspecci n empleados durante su construcci n.

Mucho m s tarde, ya entrada la edad media surgi  en Europa el sistema de organizaci n en gremios. Estos impon an los precios y especificaciones de los distin productos de los que prove an a la sociedad. Los productos de calidad daban prestigio al artesano, as  como al gremio de la zona cuando todos sus artesanos seg sus especificaciones. Este hecho constituye una de las primeras pruebas de un organismo que se encarga tanto de fijar unas normas b sicas, como de controlar s cumplimiento.

Con la revoluci n industrial comienza a desaparecer el artesanado, se crean grandes organizaciones y los antiguos artesanos se transforman en los trabajadores c empresas. En esta  poca Taylor elabor  su teor a acerca de la "gesti n cient fica del trabajo", cuyo objeto fue la preparaci n de normas para que los trabajadores cumpliesen. Comenz  con ello la instauraci n paulatina de la divisi n del trabajo, lo que supon a que los operarios interviniesen solamente en algunas operacio proceso productivo. Este hecho provoc  la necesidad de que surgiese la figura de los empleados dedicados a tareas de inspecci n, aunque se prestaba m s atenc forma de realizar el trabajo (los procesos) que a la calidad de los productos.

Finalmente el control de calidad moderno o control de calidad estad stico comenz  en los a os 30 del siglo XX con la aplicaci n industrial del cuadro de contri ideado por el Dr. W. A. Shewhart, de Bell Laboratories, que fue el inventor de los conocidos gr ficos de control.

Continuando en este proceso cronol gico destaca el hecho de que pasados unos a os del final de la II Guerra Mundial los japoneses comienzan a hacer verdade r  nfasis en la calidad. En 1950 la Uni n de Cient ficos e Ingenieros Japoneses realiz  un seminario cuyo conferenciante, el Dr. W.E. Deming, desarroll  los sig

temas: Cómo mejorar la calidad mediante el ciclo de planear, hacer, verificar y actuar; La importancia de captar la dispersión en las estadísticas; Control de proceso mediante el empleo de cuadros de control y su aplicación.

Cuatro años más tarde el Dr. J.M. Juran introdujo en Japón la idea de que la calidad de un producto o servicio residía en el grado de mentalización de todo el personal de la organización e impartió seminarios a los mandos altos y medios de las empresas niponas, explicándoles las funciones que les correspondían a cada uno en la promoción del control de calidad.

La visita de Juran marcó una transición en las actividades de control de calidad y creó un ambiente en que se reconoció el Control de Calidad como un instrumento de gerencia abriéndose las puertas para el establecimiento del control total de calidad tal como se concibe hoy.

4. Calidad: definiciones.

Una vez revisados los antecedentes del concepto calidad para concretar su significado se van a reproducir tres definiciones de calidad emanadas de personas y entidades de reconocido prestigio:

- "Conjunto de esfuerzos efectivos de los diferentes grupos de una organización para la integración del desarrollo, del mantenimiento y de la superación de la calidad de un producto, con el fin de hacer posible la fabricación y servicio a satisfacción completa del consumidor y al nivel más económico" [Feigenbaum, Deming y Juran]
- "La mejor calidad que una empresa puede producir con su tecnología de producción y capacidades de proceso actuales, y que satisfará las necesidades de clientes, en función de factores tales como el coste y el uso previsto" [Dr. Kaoru Ishikawa]
- "La gestión de calidad en la empresa es el proceso de identificar, aceptar, satisfacer y superar constantemente las expectativas y necesidades de todos los colectivos humanos relacionados con ella, clientes, empleados, directivos, propietarios, proveedores y la comunidad con respecto a los productos y servicios que esta proporciona" [consultora Arthur Andersen]

De todas estas definiciones se extraen una serie de parámetros básicos que definen la calidad: si se desea producir productos y servicios de buena calidad para el consumidor será necesario decidir por adelantado que calidad de producto (o servicio) planificar (calidad de diseño), producir (calidad de fabricación) y vender (calidad que desea el cliente).

5. Calidad del software.

A la hora de definir la calidad del software se debe diferenciar entre la calidad del producto software y la calidad del proceso de desarrollo de éste (calidad de diseño y fabricación). No obstante, las metas que se establezcan para la calidad del producto van a determinar los objetivos a establecer de calidad del proceso de desarrollo ya que la calidad del primero va a depender, entre otros aspectos, de ésta. Sin un buen proceso de desarrollo es casi imposible obtener un buen producto. Este proceso constituye el objeto del presente trabajo.

Pero la calidad del producto software se diferencia de la calidad de otros productos de fabricación industrial, ya que el software tiene sus propias características específicas:

- El software es un producto mental, no restringido por las leyes de la Física o por los límites de los procesos de fabricación. Es algo abstracto, un intangible.
- Se desarrolla, no se fabrica. El coste está fundamentalmente en el proceso de diseño, no en la posterior producción en serie, y los errores se introducen tempranamente en el diseño, no en la producción.
- Los costes del desarrollo de software se concentran en las tareas de Ingeniería, mientras que en la fabricación clásica los costes se acentúan más en las tareas de producción.
- El software no se deteriora con el tiempo. No es susceptible de los efectos del entorno y su curva de fallos es muy diferente de la del hardware. Todos los problemas que surjan durante el mantenimiento estaban allí desde el principio y afectan a todas las copias del mismo; no se generan nuevos errores.
- Es artesanal en gran medida. El software, en su mayoría, se construye a medida, en vez de ser construido ensamblando componentes existentes y ya probados, lo que dificulta aún más el control de su calidad.
- El mantenimiento del software es mucho más complejo que el mantenimiento del hardware. Cuando un componente del hardware se deteriora se sustituye una pieza de repuesto, pero cada fallo en el software implica un error en el diseño o en el proceso mediante el cual se tradujo el diseño en código máquina ejecutable.
- Es engañosamente fácil realizar cambios sobre un producto software, pero los efectos de estos cambios se pueden propagar de forma explosiva e incontrastablemente.
- Como disciplina, el desarrollo de software es aún muy joven, por lo que las técnicas de las que dispone aún no están perfeccionadas.
- El software con errores no se rechaza. Se asume que es inevitable que el software presente algunos errores de poca importancia.

También es importante destacar que la calidad de un producto software debe ser considerada en todos sus estados de evolución (especificaciones, diseño, código fuente y ejecutable). No basta con verificar la calidad del producto una vez finalizado cuando los problemas de mala calidad ya no tienen solución o su reparación es muy costosa.

La problemática general a la que se enfrenta el software es:

- Aumento constante del tamaño y complejidad de los programas.
- Carácter dinámico e iterativo a lo largo de su ciclo de vida, es decir que los programas de software a lo largo de su vida cambian o evolucionan de una versión a otra para mejorar las prestaciones con respecto a las anteriores.
- Dificultad de conseguir productos totalmente depurados, ya que en ningún caso un programa será perfecto.
- Se dedican elevados recursos monetarios a su mantenimiento, debido a la dificultad que los proyectos de software entrañan y a la falta de normalización a la hora de realizar los proyectos.
- No suelen estar terminados en los plazos previstos, ni con los costes estipulados, ni cumpliendo los niveles deseables de los requisitos especificados por el usuario.
- Incrementos constantes de los costes de desarrollo debido entre otros, a unos niveles de productividad bajos.
- Los clientes tienen una alta dependencia de sus proveedores por ser en muchos casos aplicaciones a "medida".
- Procesos artesanales de producción con escasez de herramientas.
- Insuficientes procedimientos normalizados para estipular y evaluar la productividad, costes, y calidad.

Todo lo anterior puede concretarse en:

- Ausencia de especificaciones completas, coherentes y precisas previas por parte del cliente, así como posteriores por parte de los proveedores del software.
- Ausencia de la aplicación sistemática de métodos, procedimientos y normas de ingeniería del software.
- Escasez o ausencia de entornos integrados de programación.
- Escasez de uso de técnicas actuales y automatizadas para la gestión de proyectos.
- Escasez de personal con formación y experiencia en los nuevos métodos, normas y uso de entornos y utilidades de programación.
- Otros derivados del grado de desarrollo técnico y organizativo de cada compañía.

6. La calidad a través de la normalización en la ingeniería del software y su problemática.

La normalización consiste en un proceso donde se elaboran guías, normas y convenciones sobre una determinada materia, con el objeto de definir, simplificar y estandarizar las actividades relacionadas con la materia de que se trate.

La Ingeniería del Software (IS) se ha ido desarrollando en los últimos 15 años, a través de la creación e implantación en la industria software de métodos, procedimientos, técnicas y útiles que tratan de cubrir las necesidades de cada una de las etapas del ciclo de vida de un producto software, desde la definición de requisitos hasta su mantenimiento una vez el producto comience a emplearse. Y ello con las restricciones generales de todos los procesos modernos de ingeniería: la necesidad creciente de incrementar la productividad de la programación mejorando y garantizando, simultáneamente la calidad del producto resultante.

La creación e implantación de normas de desarrollo del software son un auténtico desafío que tiene la IS como medio de comunicación para transferir sus métodos, técnicas y procedimientos a la industria del software para el diseño y desarrollo de nuevos productos. Estas normas tienen como criterio general de desarrollo

maximizar la comunicación entre los profesionales del software a través de la definición de **documentos** generales que se han de producir, proveyendo de guías indican a nivel de detalle el contenido de dichos documentos y recomendaciones de las actividades que hay que realizar durante todo el proceso de producción de software. En pocas palabras, las normas de IS son la solución a una de las mayores necesidades de la industria del software actual: la comunicación mas adecuada precisa entre sus profesionales.

A medida que ha ido aumentando la necesidad de un software más fiable, se ha reconocido que las normas de ingeniería del software (NIS) son una contribución fundamental para asegurar la producción de software de calidad. Además una consecuencia del **objetivo** genérico de mejorar **la comunicación** es que se reducen costes por un aumento de productividad y una mejora de la calidad de los desarrollos de software.

En relación a las normas los profesionales se encuentran con un problema fundamental: la dispersión de las normas relativas al software que, con frecuencia, han sido creadas por organismos muy diversos, bajo enfoques distintos y destinadas a ámbitos de actuación diferentes. Muchas compañías, por su parte, se han visto obligadas a generar sus propias normas cuando no disponen de una norma general. De hecho muchas organizaciones desarrollan sus propios **conjuntos** de normas adecuándolas a sus fines específicos. Pero también se dan casos en que organizaciones distintas tienen los mismos objetivos por lo que resultaría razonable su colaboración y, en todo caso, la **adopción** de las normas de **la organización** que tenga más avanzados sus desarrollos y un ámbito de actuación más amplio. Puede afirmarse que en la actualidad se ha llegado a un nivel de madurez en la industria del software que ha permitido a todos los implicados que exista un interés por aunar sus experiencias y esfuerzos para crear normas generales que abarquen sus áreas de interés.

Estos esfuerzos varían en cuanto al tipo de **industrias** o usuarios así como en lo relativo a los logros alcanzados, pero la tendencia actual es hacia la normalización del proceso de desarrollo de software a través de normas que conduzcan a homogeneizar los planes de garantía de calidad de él, los planes de gestión de la configuración de software, la **documentación** de sus pruebas, etc.

De hecho en EEUU existen varias entidades públicas y privadas que publican normas en diversos aspectos del software entre las que destacan: el American Institute of Aeronautics and Astronautics (AIAA), la American Nuclear Society (ANS), la American Society of Quality Control (ASQC), la Data Processing Management Association (DPMA) y la Electronic Industries Association (EIA).

Sin embargo las actividades de normalización en IS son relativamente recientes, siendo tres las **instituciones** de mayor prestigio y difusión que las generan: el Instituto de Ingenieros Eléctricos y Electrónicos de los EE.UU. (IEEE) a través del Subcomité de Normas de IS de la Computer Society, el National Bureau of Standards (NBS) que publica las normas FIPS (Federal Information Processing Standards) o normas a aplicar en todos los Estados Federales y el Departamento de Defensa (DOD -USAF, ARMY y NAVY-) que publica normas en el terreno militar.

En cuanto al proceso de creación e implantación de las normas, nos remitiremos a las dictadas por el IEEE: una nueva **idea** puede tenerla y sugerirla cualquiera de sus miembros, pasándola a continuación al Comité de Normas de IS, el cual cursará una Petición de Autorización de **Proyecto** (PAP) que se transmite a todos los grupos que hacen normas y al ANSI (American National Standard Institute) que se encargará de la **coordinación**.

La PAP define el objetivo, ámbito, **descripción** y principales contactos para llevar a cabo el proyecto. La etapa siguiente corresponde al desarrollo de la norma a cargo de un **grupo** de trabajo. Cualquiera puede adherirse al grupo y el IEEE se encargará de implicar a todos los interesados para que participen en reuniones con todos los demás miembros (unas cuatro reuniones anuales). También pueden intervenir a distancia detallando las contribuciones oportunas y remitiéndolas al grupo correspondiente. El proceso será más enriquecedor cuantos más profesionales de distinto nivel y más organismos intervengan.

Una vez terminado un desarrollo se concretará en un documento (borrador o draft) que pasará por varias revisiones como consecuencia del proceso de valoración y votación para aceptarlo. Durante este periodo habrán de votar al menos el 75% de los miembros del grupo de votación (unos 100 expertos) y a su vez el 75% de los miembros del grupo de votación. Tras su aprobación se distribuirá la nueva norma. En cuanto al tiempo medio que se tarda desde que se inicia la elaboración de una norma hasta que se aprueba es de tres años. Finalmente, como trabajo complementario se procede a difundir la norma a través de Seminarios.

La etapa siguiente en este ciclo corresponde con la de uso de la norma por las empresas u organismos. Durante esta fase los usuarios proporcionan comentarios positivos o negativos acerca de la norma, según su propia experiencia, y en ocasiones aceptan o modifican las normas para sus fines específicos. Las modificaciones remiten al consejo de normas del IEEE para su **evaluación** por el grupo de trabajo entrando en una nueva fase del ciclo de elaboración de normas. Finalmente se decidirá dejarla igual, modificarla o eliminarla si ya ha sido superada por una nueva norma. El ciclo completo dura cinco años.

7. Los modelos de calidad del software de gestión.

En este apartado se presenta, como propuesta de ayuda para mejorar la calidad del software de gestión, una **síntesis** del modelo de McCall por ser uno de los más difundidos y porque además ha servido de base para otros modelos (el modelo de Boehm y el Software Quality Management -SQM- de Murine).

En general los modelos de calidad definen a ésta de forma jerárquica, o sea la calidad se produce como consecuencia de la evaluación de un conjunto de **indicadores** métricas en diferentes etapas:

En el nivel más alto de la jerarquía se encuentran los factores de calidad definidos a partir de la visión del usuario del software, y conocidos también como atributos de calidad externos.

Cada uno de los factores se descompone en un conjunto de criterios de calidad, o sea aquéllos atributos que cuando están presentes contribuyen a obtener un software de alta calidad. Se trata de una visión de la calidad técnica, desde el **punto** de vista del producto software y se les denomina también atributos de calidad internos.

Finalmente para cada uno de los criterios de calidad se definen un conjunto de métricas o medidas cuantitativas de ciertas características del producto que indican el grado en que dicho producto posee un determinado atributo de calidad.

De esta manera, a través de un modelo de calidad se concretan los aspectos relacionados con ella de tal manera que se puede definir, medir y planificar. Además el empleo de un modelo de calidad permite comprender las relaciones que existen entre diferentes características de un producto software.

En contra de los modelos de calidad pesa que aún no ha quedado demostrada la validez absoluta de ninguno de ellos.

8. El modelo de McCall.

El modelo de McCall organiza los factores en tres ejes o puntos de vista desde los cuales el usuario puede contemplar la calidad de un producto, basándose en cinco factores de calidad organizados en **torno** a los tres ejes y a su vez cada factor se desglosa en otros criterios:

Puntos De Vista O Ejes	Factor	Criterios
OPERACIÓN DEL PRODUCTO	Facilidad de uso	- Facilidad de operación: Atributos del software que determinan la facilidad de operación del software. - Facilidad de comunicación: Atributos del software que proporcionan entradas y salidas fácilmente asimilables. - Facilidad de aprendizaje : Atributos del software que facilitan la familiarización inicial del usuario con el software y la transición del modo actual a la operación. - Formación: El grado en que el software ayuda para permitir que nuevos usuarios apliquen el sistema.
	Integridad	- Control de accesos. Atributos del software que proporcionan control de acceso al software y los datos que maneja. - Facilidad de auditoría : Atributos del software que facilitan la auditoría de los accesos al software. - Seguridad : La disponibilidad de mecanismos que controlen o protejan los programas o los datos.
	Corrección	- Completitud: Atributos del software que proporcionan la implementación completa de todas las funciones requeridas.

		<ul style="list-style-type: none"> - Consistencia: Atributos del software que proporcionan uniformidad en las técnicas y notaciones de diseño e implementación. - Trazabilidad o rastreabilidad: Atributos del software que proporcionan una traza desde los requisitos a la implementación con respecto a un operativo concreto.
OPERACIÓN DEL PRODUCTO	Fiabilidad	<ul style="list-style-type: none"> - Precisión: Atributos del software que proporcionan el grado de precisión requerido en los cálculos y los resultados. - Consistencia. - Tolerancia a fallos: Atributos del software que posibilitan la continuidad del funcionamiento bajo condiciones no usuales. - Modularidad: Atributos del software que proporcionan una estructura de módulos altamente independientes. - Simplicidad: Atributos del software que posibilitan la implementación de funciones de la forma más comprensible posible. - Exactitud: La precisión de los cálculos y del control.
	Eficiencia	<ul style="list-style-type: none"> - Eficiencia en ejecución: Atributos del software que minimizan el tiempo de procesamiento. - Eficiencia en almacenamiento: Atributos del software que minimizan el espacio de almacenamiento necesario.
REVISIÓN DEL PRODUCTO	Facilidad de mantenimiento	<ul style="list-style-type: none"> - Modularidad. - Simplicidad. - Consistencia. - Concisión: Atributos del software que posibilitan la implementación de una función con la menor cantidad de códigos posible. - Auto descripción: Atributos del software que proporcionan explicaciones sobre la implementación de las funciones.
	Facilidad de prueba	<ul style="list-style-type: none"> - Modularidad. - Simplicidad. - Auto descripción. - Instrumentación: Atributos del software que posibilitan la observación del comportamiento del software durante su ejecución para facilitar las mediciones del uso o la identificación de errores.
	Flexibilidad	<ul style="list-style-type: none"> - Auto descripción. - Capacidad de expansión: Atributos del software que posibilitan la expansión del software en cuanto a capacidades funcionales y datos. - Generalidad: Atributos del software que proporcionan amplitud a las funciones implementadas. - Modularidad.
	Reusabilidad	<ul style="list-style-type: none"> - Auto descripción. - Generalidad. - Modularidad. - Independencia entre sistema y software: Atributos del software que determinan su dependencia del entorno operativo. - Independencia del hardware: Atributos del software que determinan su dependencia del hardware.
	Interoperabilidad	<ul style="list-style-type: none"> - Modularidad. - Compatibilidad de comunicaciones: Atributos del software que posibilitan el uso de protocolos de comunicación e interfaces estándar. - Compatibilidad de datos: Atributos del software que posibilitan el uso representaciones de datos estándar. - Estandarización en los datos: El uso de estructuras de datos y de tipos estándar a lo largo de todo el programa.
	Portabilidad	<ul style="list-style-type: none"> - Auto descripción. - Modularidad. - Independencia entre sistema y software. - Independencia del hardware.

9. Cómo emplear el modelo de mcall.

Antes de comenzar a utilizar el modelo de McCall hay que seguir las siguientes pautas:

1. Se aceptan los factores, criterios y métricas que propone el modelo.
2. Se aceptan las relaciones entre factores y criterios, y entre criterios y métricas.
3. Se selecciona un subconjunto de factores de calidad sobre los que aplicar los requisitos de calidad establecidos para el proyecto.

Al comienzo del proyecto habrá que especificar los requisitos de calidad del producto software, para lo cual se seleccionarán los aspectos inherentes a la calidad deseada del producto, teniendo que considerarse para ello:

- Las características particulares del propio producto que se está diseñando: por ejemplo, su ciclo de vida que si se espera que sea largo implicará un mayor énfasis en la facilidad de mantenimiento y la flexibilidad, o bien si el sistema en desarrollo está destinado a un entorno donde el hardware evoluciona rápidamente implicará como requisito su portabilidad, ...
- La relación calidad-**precio**, que puede evaluarse a través del coste de cada factor de calidad frente al beneficio que proporciona. La siguiente tabla **muestra** relación calidad-precio para cada factor considerado:

Factor	Beneficio / coste
Corrección	alto
Fiabilidad	alto

Boehm, B.W., Brown, J.R., Lipow, M.; Quantitative Evaluation of Software Quality, Proceedings 2nd International Conference on Software Engineering, pp. 605, 1976.

Cavano, J.P., McCall, J.A., A Framework for the Measurement of Software Quality, Proc. of the ACM Software Quality Assurance Workshop, pp. 133-139, 1978.

Chidung Lac; Raffy, J.-L., A tool for software Quality, Proceedings of the Second Symposium on Assessment of Quality Software Development Tools; New Orleans, LA, USA; 27-29 May 1992; IEEE Comput. Soc. Press; pp. 144-150; Nahouraii, E. (ed.).

De Domingo, J. y Arranz, A., Calidad y mejora continua, Ed Donostiarra. 1997.

De Millo, R. A. et al., Software Testing and Evaluation, Benjamin/Cummings Pub. Co., 1987.

Dijkstra, E.W., Formal development of programs and proofs, Addison-Wesley, 1989.

Hivart, M.P.; Romain, M.M.; Software Quality measurement in complex systems, Proceedings 7th International Conference on Reliability and Maintainability France; pp. 18-22, Jun. 1990.

Hoare, C.A.R., An Axiomatic Basis for Computer Programming, Communications of the ACM 12, 10, pp. 576-583, Oct. 1969.

Howden, W.E., Reliability of the Path Analysis Testing Strategy, IEEE Transactions on Software Engineering SE-2, 3 (Sept. 1976), pp. 37-44, 1976.

Kitchenham, B.; Towards a Constructive Quality Model, Software Engineering Journal, Vol.2, N. 4, pp. 105-113, 1987.

Miller, E., Howden, W. E., Tutorial, Software Testing & Validation Techniques, 2a ed., IEEE Computer Society Press, 1981.

Murine, G.E., Integrating software quality metrics with software QA, Quality Progress vol.21, no.11; pp. 38-43; Nov. 1988.

Oman, P. W., A case study in SQA Audits, Software Quality Journal nº 2, pp. 13-27, 1993.

Pressman, Roger S, Ingeniería del software, un enfoque práctico, McGraw Hill 95.

Rapps, S., Weyuker, E.J., Selecting Software Test Data Using Data Flow Information, IEEE Transactions on Software Engineering SE-11, 4 (Abr. 1985), pp. 375, 1985.

Reifer, D. J., Knudson, R. W., Smith, J., Final report : Software Quality Survey, American Society for Quality Control, Aerospace Industries Association of America, 1988.

Richards Adrion, W., Branstad M.A., Cherniavsky, J.C., Validation, Verification and Testing of Computer Software, Computing Surveys, Vol. 14, Nº 2, pp. 119-192, Junio 1982.

Senn, James, Análisis y diseño de sistemas de información, McGraw Hill 1997.

RESUMEN: La mayor importancia de las nuevas tecnologías de la información y su creciente presencia en los diversos ámbitos de la industria moderna (robo, centros de control, etc.) y sus productos finales (aviación, automóviles, electrodomésticos, telefonía, etc.) conlleva cada vez más la presencia de programas informáticos que gobiernan muchas de sus prestaciones, o bien como herramientas que el cliente empleará en su propio beneficio. Basta con observar la frenética actividad que ha supuesto para la industria, las empresas de servicios y la Administración el enfrentarse al tan temido "efecto 2000", así como el gasto que ha conllevado la revisión y modificación de los programas, para vislumbrar la punta de un iceberg: la falta de un control riguroso y sistemático de la calidad del software de gestión. En el presente trabajo se aborda este tema y se presenta un modelo de aplicación que ayudaría a proveedores y clientes desde el comienzo el diseño de una aplicación específica de software para su negocio o actividad.

PALABRAS CLAVES: calidad, hardware, modelo de McCall, software.

Autor:

Cervera Paz, Angel

Dpto. Organización de Empresas
Núñez Moraleda, Bernardo M.;
Dpto. Lenguajes y Sist. Informáticos
Universidad Cádiz

Comentarios

Para dejar un comentario, [regístrate gratis](#) o si ya estás registrado, [inicie sesión](#).

Trabajos relacionados

[Estudio sobre los lenguajes de programación para la robótica](#)

Origen de la palabra robot y su significado. Propiedades características de los robots. El robot y su funcionamiento. Cl...

Estructura de un objeto. Encapsulamiento y ocultación. Organización de los objetos. Actualmente una de las áreas más ca...

[Sistemas de Procesamiento de Datos Programación Orientada a Objetos](#)

[Rupturas de Informe](#) Definición de una Ruptura de Informe.

Especificación de Opciones de Proceso. Una Ruptura de Informe se usa para dividir...

Ver mas trabajos de [Programación](#)

Nota al lector: es posible que esta página no contenga todos los componentes del trabajo original (pies de página, avanzadas formulas matemáticas, esquemas o tablas complejas, etc.). Recuerda para ver el trabajo en su versión original completa, puede descargarlo desde el [menú superior](#).

Todos los documentos disponibles en este sitio expresan los puntos de vista de sus respectivos autores y no de Monografias.com. El objetivo de Monografias.com es poner el conocimiento a disposición de toda su comunidad. Queda bajo la responsabilidad de cada lector el eventual uso que se le de a esta información. Asimismo, es obligatoria la cita del autor del contenido y de Monografias.com fuentes de información.

